Framework for Constructing Context-Specific Migration Methods for Test Cases

Presented by Ivan Jovanovikj
Agenda

1. Challenges in Test Case Reuse in Migration Projects

2. Construction of Context-Specific Migration Methods

3. Outlook and Conclusion
1. CHALLENGES IN TEST CASE REUSE IN MIGRATION PROJECTS
Problem Statement

How to systematically reuse existing test cases in a software migration scenario?

Test the desired functionality of the source system

Test Cases

System

Model

Restructuring

Migrated Model

Changes from the software migration must be considered

Forward Engineering

Test Requirements must be considered

Software Migration

Test Case Design is costly and time consuming

Test Cases

User Conference on Advanced Automated Testing
Solution Approach

In this talk:
Framework for Constructing Context-Specific Migration Methods for Test Cases

Solution Approach
Development of a Context-Specific Test Case Migration Method

System

Software Migration

Migrated System

Test Cases

Migrated Test Cases
2. Construction of Context-Specific Migration Methods
Approach Overview

Pre-Migration
- Migration Context Characterization
  - Context Model
  - Reference Migration Method

Migration
- Migration Method Tailoring
  - Adapted Migration Method
- Tool Implementation
- Migration Method Enactment
  - Context-specific Tools

Post-Migration
- Migration Validation
Migration Context Characterization

System Migration Context
1. What was migrated?
2. How was it migrated?

Testing Context
3. What is the current state of the test cases?
4. What are the testing requirements?
Migration Context Factors

System Migration Context Factors
- Source Environment Characteristics
  - Language
  - Framework
  - Architecture
- Target Environment Characteristics
  - Language
  - Framework
  - Architecture
- Transformation Characteristics
  - Degree of Automation
  - Degree of Formality
  - Process Support
  - Change Type

Testing Context Factors
- Source Testing Environment Characteristics
  - Testing Framework
  - Test Case Type
  - Test Case Anatomy
  - Test Case Quality
- Target Testing Environment Characteristics
  - Testing Framework
  - Test Case Type
  - Test Case Anatomy
  - Test Case Quality
System Migration Context - Case Study

Eclipse Modeling Project

Eclipse Modelling Framework
- EMF Ecore
- EMF.Edit
- EMF.CodeGen
- Just-in-Time Compilation (JIT)

Model Development Tools
- OCL
- BPMN2
- MoDisco

Modelling Framework
-.ecore
- OCL
- Ahead-of-Time Compilation (AOT)

Migration Methods
- Tool Implementation
- Migration Method Enactment
- Migration Validation

User Conference on Advanced Automated Testing

© All rights reserved
Testing Context - Case Study

Source Testing Environment Characteristics

- Test Cases
  - OCL Test Framework
    - Testing Framework (JUnit)
    - Language (Java)

Target Testing Environment Characteristics

- Test Cases
  - OCL Test Framework
    - Testing Framework (NUnit)
    - Language (C#)

1. Migration of TestOCL
2. Migration of Test Cases
Reference Migration Method

- **UTP Meta-Model**
- **Platform Independent Test Model**
  - **Test Abstraction**
  - **Test Case Understanding**
- **Platform Specific Test Model (Source)**
  - **Test Abstraction**
  - **Test Case Understanding**
  - **Framework Transformation**
- **Model of the Code**
  - **Model Discovery**
- **Platform Specific Test Model (Migrated)**
  - **Test Concretization**
- **Model of the Migrated Code**
  - **Language Transformation**
  - **Code Generation**
- **Test Code**
  - **Reimplementation**
- **Migrated Test Code**

**Activities**
- Text to Text Transformation
- Text to Model / Model to Text Transformation
- Model to Model Transformation
- Exogeneous Transformation
- Endogeneous Transformation

**Artifacts**
- Textual Artifact
- Model
- Meta Model
Method Tailoring

- Reductive Tailoring
- Remove Activity
- Remove Artifact

Platform Specific Test Model (Source) → Test Abstraction → Test Case Understanding → Model of the Code → Model Discovery → Test Code


Platform Specific Test Model (Migrated) → Test Concretization

Restructuring

Test Concretization → Code Generation → Migrated Test Code

Reimplementation

User Conference on Advanced Automated Testing
Method Tailoring

Context C1 specific
Context C2 specific
Context C3 specific
Context C4 specific
Context C5 specific
Context C6 specific
Context C7 specific
Context C8 specific
Context C9 specific
Context C10 specific
Context Cn specific
Method Tailoring

Test Case Migration Method

[Diagram showing the process of test case migration with arrows indicating steps such as Test Abstraction, Test Concretization, Framework Transformation, Model Discovery, and Test Case Understanding. Key elements include Platform Specific Test Model (Source) and (Migrated), Model of the Code, and JUnit and NUnit test frameworks.]
Method Tailoring: Test Case Migration Method

- Reductive Tailoring
- Remove Activity
- Remove Artifact

Diagram:
- Platform Specific Test Model (Source)
- Framework Transformation
- Platform Specific Test Model (Migrated)
- Test Case Understanding
- Model of the Code
- Model Discovery
- Test Code
- Reimplementation
- Migrated Test Code

Process:
- Migration Context Characterization
- Migration Method Tailoring
- Tool Implementation
- Migration Method Enactment
- Migration Validation
Method Tailoring: TestOCL Migration Method

Reductive Tailoring
- Remove Activity
- Remove Artifact

Model of the Code
- Test Case Understanding
- Model Discovery

Platform Specific Test Model (Source)
- Test Abstraction
- Framework Transformation

Test Independent Test Model
- Restructuring

Platform Specific Test Model (Migrated)
- Test Concretization
- Language Transformation

Model of the Migrated Code
- Code Generation

TestOCL Code
- Reimplementation

Migrated TestOCL Code

Java

C#
Method Tailoring: TestOCL Migration Method

Reductive Tailoring

- Remove Activity
- Remove Artifact

Java

TestOCL Code → Reimplementation → Migrated TestOCL Code

C#
Method Tailoring

2. Test Case Migration Method
   - Platform Specific Test Model (Source)
   - Framework Transformation
   - Platform Specific Test Model (Migrated)
   - Test Concretization + Code Generation
   - Migrated Test Code
   - Reimplementation
   - Test Case Understanding
   - Model Discovery
   - Model of the Code
   - Test Code
   - Reimplementation
   - C#

1. TestOCL Migration Method
   - Java
   - TestOCL Code
   - Reimplementation
   - Migrated TestOCL Code
Tool Implementation

- TestModel2 TestCase
- TestModel
- Platform Specific Test Model (Source)
- Model of the Code
- Test Code

- Framework Transformation
- Test Case Understanding
- Model Discovery
- Reimplmentation

- Platform Specific Test Model (Migrated)
- Migrated Test Code

- Test Concretization + Code Generation

- Supports

- JDT Core
- eclipse plugin
- eclipse plugin
- xtend

User Conference on Advanced Automated Testing
Migration Method Enactment

1. TestOCL Migration
   - Java TestOCL Code
   - C# Migrated TestOCL Code
   - Reimplementation

2. Test Case Migration
   - Platform Specific Test Model (Source)
   - Framework Transformation
   - Platform Specific Test Model (Migrated)
   - Test Concretization
     + Code Generation
   - Migrated Test Code
   - Reimplementation

Tools
- eclipse plugin
- TestModel2
- TestCase2

User Conference on Advanced Automated Testing
Migration Method Enactment

- Migration Method Enactment
- Migration Context Characterization
- Tool Implementation
- Migration Method Tailoring
- Migration Method Enactment
- Migration Validation

Diagram:
- Eclipse plugin
- TestCase2 TestModel
- Platform Specific Test Model (Source)
- Test Case Understanding
- Model Discovery
- Model of the Code
- Test Code
- Framework Transformation
- Platform Specific Test Model (Migrated)
- Test Concretization + Code Generation
- Migrated Test Code
- Reimplemenation
- Eclipse plugin
- TestModel2 TesCase
- Xtend

User Conference on
Advanced Automated Testing

© All rights reserved
Migration Method Enactment

- **迁移方法实现**

  - **_TestCase2** → **TestModel2**
  - **xUnit model** → **xUnit model**
  - **JUnit** → **JUnit**
  - **Java AST** → **Java AST**

- **工具实施**
  - **Migration Context Characterization**
  - **Migration Method Tailoring**
  - **Tool Implementation**
  - **Migration Method Enactment**
  - **Migration Validation**
Migration Method Enactment

Test Case Understanding

JUnit

@Test public void testCollectionAsBag() {
    assertQueryResults(null,"Bag{1,2.0, ‘3’}" + "Sequence{1, 2.0, ‘3’} asBag()");

JUnit

xUnit model

Reimplementation

Test Concretization +Code Generation

Framework Transformation

Migration Method Tailoring

Migration Method Characterization

Migration Context Characterization

Tool Implementation

Migration Method Enactment

Migration Validation

User Conference on Advanced Automated Testing

© All rights reserved
Migration Method Enactment

- Test Case
- Understanding
- xUnit model
- Migration Method Enactment
- TestModel
- TestCase
- JDT Core plugin
- JUnit plugin
- xUnit model
- Test Concretization
- Code Generation
- Reimplementation
- Model Discovery
- Test Model
- Test Case2
- Test Model2
- AST
- Migration Context Characterization
- Migration Method Tailoring
- Tool Implementation
- Migration Method Enactment
- Migration Validation

User Conference on Advanced Automated Testing

© All rights reserved
Migration Method Enactment

- [TestCase] testCollectionAsBag
  - /eContainer
    - name = testCollectionAsBag
  - assertions (4)
    - [Assertion] assertQueryResults
      - /eContainer
        - name = assertQueryResults
        - type = assertQueryResults
      - expectedValue (1)
        - [ExpectedValue] "Bag(1, 2.0, '3')"
      - action (1)
        - [Action] "OrderedSet(1, 2.0, '3')\rightarrow asBag()"

- Test Model 2
- Test Case

- Framework Transformation

- xUnit model

- Test Case Understanding

- JUnit

- Model Discovery

- Reimplementation

- Tool Implementation
Migration Method Enactment

- **JUnit plugin**
- **TestModel**
- **TestCase2**
- **JDT Core plugin**
- **xUnit model**
- **Java AST**
- **Test Case Understanding**
- **Model Discovery**
- **Reimplementation**
- **Test Concretization**
- **Code Generation**
- **Framework Transformation**

User Conference on Advanced Automated Testing
Migration Method Enactment

Migration Method Tailoring

Migration Method Characterization

Migration Method Enactment

Migration Method Validation

Migration Validation

TestCase2
TestModel

eclipse plugin

TestModel2
tesCase

eclipse plugin

ejDT Core

JUnit

JUnit

Java AST

xUnit model

Test Case Understanding

Model Discovery

Reimplementation

Framework Transformation

Transformation

[TestMethod]
public void testCollectionAsBag_nonEmptySequence()
{
    // Arrange
    Sequence<object> initial = new Sequence<object>() { 1, 2, 0, '3' };
    Bag<object> expected = new Bag<object>() { 1, 2, 0, '3' };
    // Act
    Bag<object> actual = inital.asBag();
    // Assert
    assertQueryResults(expected, actual);
}
Migration Validation

How to validate a test case migration?

What is a valid test case migration?

Test case migration is a process of transferring test cases into new environments without changing their functionality, i.e., without changing what they test.

Behavioral Equivalence

How to ensure behavioral equivalence in test case migration?
Migration Validation

Test Cases → Migration → Migrated Test Cases → Execution → Test Report

Passed Test Cases (negatives)

- True negatives
- False negatives

Failed Test Cases (positives)

- True positives
- False positives

Problem: Detects non-existing errors
Consequence: Waste of time by looking for non-existing error

Problem: Hides potential errors
Consequence: System malfunctioning/crashes

Migrated system works as expected
Errors in the migrated system
Migration Validation

How to ensure behavioral equivalence in test case migration?

How to avoid/detect false positives and false negatives?

Constructive approaches

Analytical Approaches

Bi-simulation

Mutation Testing

Bottom Line

Cost of Test Case Migration + Cost of Validation of Test Case Migration ≤ Cost of Developing New Test Cases

User Conference on Advanced Automated Testing
OUTLOOK AND CONCLUSION
Overview and Conclusion

Migration Domain

Problem Statement
How to systematically reuse existing test cases in a software migration scenario?

Test Case Migration

Software Migration

Test Requirements must be considered

Changes from the software migration must be considered

Test Case Design is costly and time consuming

System

Migrated System
test
test
Migrated Test Cases

Test the desired functionality of the source system

System Migration Context

Java

Eclipse Modeling Project

Modeling Framework

Model Development Tools

EMF

Ecore

EMF.Edit

OCL

BPMN2

MoDisco

Test-Run-Time Compilation (RT)

Platform Specific Test Model (Source)

Model of the Code

Test Case Transformation

Test Case Understanding

Test Concretization & Code Generation

Platform Specific Test Model (Migrated)

Tools

Test Model

TestCase

JUnit

Eclipse plugin

OCL Test Framework

Testing Framework

JUnit

Language

JUnit

Language

Comprehensive discussion on the practical application

User Conference on Advanced Automated Testing
Thank you for your attention

s-lab – Software Quality Lab
Paderborn University
Zukunftsmeile 1
33102 Paderborn

Ivan Jovanovikj
Tel.: (05251) 54 65-216
ivan.jovanovikj@s-lab.uni-paderborn.de

http://s-lab.upb.de